2010

QoSDIST: A QoS Probability Distribution Estimation Tool for Web Service Compositions

Huiyuan Zheng
Macquarie University

Jian Yang
General Research Institute for Non Ferrous Metals,Ministry of Science & Technology, China,Macquarie University

Weiliang Zhao
University of Wollongong, wzhao@uow.edu.au

2010 IEEE Asia-Pacific Services Computing Conference

QoSDIST: A QoS Probability Distribution Estimation Tool for Web Service Compositions

Huiyuan Zheng, Jian Yang, Weiliang Zhao
Department of Computing
Macquarie University
NSW, Australia
{*huiyuan.zheng, jian.yang, weiliang.zhao*}*@mq.edu.au*

***Abstract*—In this paper, a QoS DIstribution eStimation Tool (QoSDIST) is developed to estimate the QoS distributions for service compositions. QoSDIST can generate QoS probability distributions for component web services. When estimating the QoS probability distribution for a service composition, QoSDIST does not put any constraints on the representation of the QoSs of component web services, i.e., the QoS of a component web service can be in single value, discrete values with frequencies, standard statistical distribution, or any general distribution regardless of its shape, which can not be done by any existing approaches. Moreover, QoSDIST can deal with commonly used composition patterns, including loop with arbitrary exit points.**

***Keywords*-QoS; probability distribution; Web service composition**

I. INTRODUCTION

The nature of services creates the opportunity for building composite services by combining existing elementary or complex services (referred to as component services) from different enterprises and in turn offering them as high-level services or processes. QoS analysis becomes increasingly challenging and important when complex and mission critical applications are built upon services with different QoS [1]. Thus solid model and method support for QoS predication in service composition become crucial and will lay a foundation in further analysis of complexity and reliability in developing service oriented distributed applications.

It is important to estimate the QoS of a composite service at design time based on the quality of individual web services to make sure that the composition can satisfy the expectations of end users [2,3]. A web service may need to be replaced at run time if it becomes unavailable or its performance degrades too much [4]. Quite often, functionally equivalent services exist with different QoS. A comparison is therefore necessary by analysing the QoS of the composite service with different service combination options.

Two issues need to be addressed to perform QoS estimation: (1) how QoS of a component web service in a service composition can be accurately represented? (2) how QoS of a service composition can be calculated based on the QoS of its component services?

Some QoS metrics, such as response time, dynamically changes. Fixed value is not able to describe the QoS effectively. For example, two web services having the same mean or maximum QoS value may have quite different distributions. A service consumer may choose one web service over the other based on the QoS distributions. Probabilistic QoS, i.e. probability density function (PDF) of QoS, has already been used in service contracts and service selections in service compositions [5,6] and shows promising result over traditional fixed QoS value methods. Well known statistical distributions such as T-location scale distribution is used in existing probabilistic QoS research. However, real service execution data shows that a distribution of a QoS metric such as response time can be of any shape, which may not fit into any well known statistical distributions. The QoS estimation result of a service composition based on these inaccurate QoS is of course not accurate either.

Furthermore, how QoS of component services are aggregated also depends on the way the composite service is constructed. [7] provides a QoS analysis tool that can handle well known statistical distributions in limited composition patterns such as sequential, parallel. The QoS aggregation for loop structure is not fully addressed in the current QoS literature.

In the rest of the paper we will use the term *component QoS* and *composite QoS* to refer to QoS of component service and QoS of composite service respectively.

In order to overcome the problems discussed above, we develop a QoS distribution estimation tool: QoSDIST, for service compositions. A Web-based graphical user interface (WebGUI) runs on the client side. The QoSDIST server is in charge of most of the processing and provides users with the QoS estimation results for service compositions. Compared with existing methods, QoSDIST has the following contributions:

- Gaussian Kernel Density method is adopted to generate the QoS probability distributions for component web services. This method does not rely on assumptions that the data are drawn from a given probability distribution. This property makes this method more robust and accurate than existing QoS probability distribution

978-0-7695-4305-5/10 $25.00 © 2010 IEEE
DOI 10.1109/APSCC.2010.19

generation methods for component web services. With this QoS estimation approach together with a QoS probability updating method, the QoS distributions of web services can be regularly updated which makes the estimated QoS much more close to a real time QoS. By doing this, we reduce the impact of the network performance on QoS of web services.

- A QoS probability calculation approach is developed for service compositions and applied in QoSDIST. The proposed QoS estimation method does not make any assumption about the form of the component QoS, i.e., the component QoS can be of: a single value, discrete values with frequencies (i.e. probability mass function (PMF)), a well known statistical distribution, or any distribution regardless of its shape. This property brings challenges in QoS aggregation for service compositions, but at the same time makes this method more robust than existing methods. The ability of covering all QoS representation forms is important because the representation of different QoS metrics can not be the same. For example, a probability distribution is appropriate to represent response time, a single value is good to represent availability, while a probability mass function can represent cost well.

For the clarity of the research, we make the following assumptions: (1) Each QoS metric is calculated individually without considering the correlations with other QoS matrics. For example, *response time* may be correlated to *cost*. However, we will analyse and calculate the composite QoS for *response time* and *cost* separately. (2) In the composite QoS calculation, we will not consider the cases when component services seriously affected with each other on their QoS. This situation only happens when these component services are simultaneously invoked on the same server.

The remainder of the paper is organized as follows: Section II discusses the related work in composite QoS analysis and the contributions of this paper. Section III discusses the QoS probability distribution generation approaches for web services and service compositions respectively. Section IV provides experimental results to show the effectiveness of the proposed QoS probability generation method for web services and the soundness of the proposed QoS probability estimation method for service compositions. Section V gives an overview of the implementation of QoSDIST. Section VI concludes the paper.

II. Related Work

Existing research in component QoS representation can be categorized as: single values [8], probability mass functions [9], and well known statistical distributions [6].

[9] claims that it is more reasonable to model component QoS as probability mass functions (i.e. multiple values with different frequencies). In our previous work [10], an approach is proposed to calculate the QoS for every possible execution path of a composition and then a discrete QoS distribution is generated for the composition.

[8] mentions that a QoS metric can be specified as a well known statistical distribution, such as Exponential, Normal, Weibull, and Uniform. [6] argues that QoS probability distributions can be used to express contracts criteria. T location-scale distribution is adopted to fit the original monitored data (i.e. QoS samples) of Web services.

In [5], a probabilistic approach is proposed to select web services whose QoS are modelled as probability distributions. The response time of a web service is a probability distribution following no obvious pattern generated by an aggregation of Normal distributions under different work load.

For single value represented QoS, aggregation method [8] is proposed to calculate the composite QoS. A composition can be regarded as being composed of composition patterns. Formulae to calculate QoS for these patterns are given. But these formulae can only be applied to single values.

For QoS represented by discrete values with frequencies (i.e. probability mass functions) [9], the calculation method is much the same as it is for single values. The difference is that the probability of each possible QoS value of the composite service needs to be taken into account.

For the well known statistical distribution represented QoS [6], simulation approaches are applied to compute the composite QoS. The real QoS data of component services are fitted with standard statistical distributions.

[7] presents a tool for predicting composite QoS. Component QoS can be modeled as single value or well known statistical distributions. But this tool does not support complex patterns such as loop.

III. QoS Probability Distribution Estimation for Component Web Services and Service Compositions

A. QoS Probability Distribution Generation for Component Web Services

In this subsection, we will introduce the technique of generating the QoS probability distributions for web services based on their history execution logs. We adopt Gaussian Kernel Density estimation approach [11] to do the QoS generation. Compared with existing methods' fitting a QoS sample with a well known QoS probability distribution, the method used in this paper is distribution free, which does not rely on assumptions that the data are drawn from a given probability distribution. This property makes this method more robust than existing web services' QoS generation methods.

Gaussian Kernel Density estimation is a non-parametric way of estimating the probability density of a random variable. Assume that $x_1, x_2, ..., x_n$ are a set of QoS data for a specific QoS metric of a web service. These data can be drawn from the history execution log of the web service.

Then the Gaussian Kernel Density approximation of the QoS distribution is [11]:

$$\hat{f}(x) = \frac{1}{nh}\sum_{1}^{n}\frac{1}{\sqrt{2\pi}}e^{-\frac{(x-xi)^2}{2h^2}} \quad (1)$$

where h is a smoothing parameter and can be calculated as follows:

$$h = 1.06\sigma n^{-1/5} \quad (2)$$

where n is the size of the sample and σ is the standard deviation of the sample.

B. QoS Probability Distribution Estimation for Service Compositions

A composite service is regarded as being constructed based on four composition patterns, i.e. sequential, parallel, conditional, and loop (see Figure 1). The vertices in Figure 1 represent web services and the arcs represent the transitions from one web service to another. It can be seen that in a Sequential Pattern, one web service is directly followed by another one; in a Parallel Pattern, the web services in different branches split from one web service, run concurrently, and finally merge to another web service when the web services in all the branches finish running; in a Conditonal Pattern, only one of the branch can be executed; in a Loop Pattern, the web services in the pattern will be run continuously until certain condition is met. The formal definition for these patterns as well as the modelling method of these patterns can be found in our previous work [12].

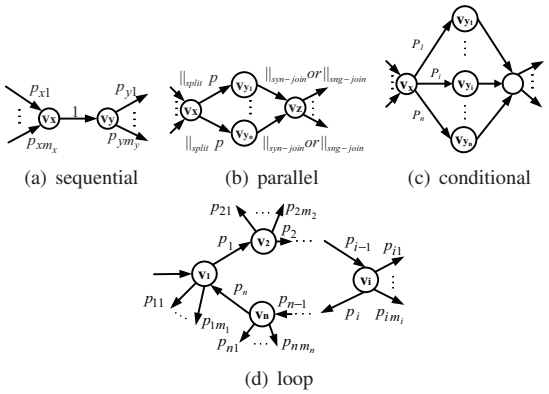


Figure 1. Basic Composition Patterns

We design a calculation approach which can compute QoS for service compositions. In this approach, we assume that the QoS of web services are independent of each other. The QoS metric for a component web service can be represented by a constant value or a probability distribution, which can either be a probability mass function, a well known statistical probability distribution, or a general probability distribution. The QoS metric response time is taken as an example here. It should be noticed that though the discussion is based on distributions of response time, the developed formulae for composition patterns are also applied to other QoS metrics and these QoS metrics can be represented by single values or probability distributions. This is because single values and probability mass functions can also be represented as distributions with the help of Dirac delta function[1]. The QoS calculation formulae for different composition patterns are listed as follows:

1) Sequential Pattern: The response time of a Sequential Pattern is the sum of the QoS of its component web services. When the component QoS is represented by probability density functions (PDF), the QoS of a Sequential Pattern is the convolution of the PDFs of the component QoS,i.e.

$$f(q) = (f_1 * f_2)(q) = \int_0^q f_1(x) f_2(q-x)dx \quad (3)$$

where $f(q)$ is the PDF of response time of a Sequential Pattern, $f_1(q)$ and $f_2(q)$ are the PDFs of the component QoS.

2) Parallel Pattern: The response time of a Parallel Pattern with synchronized merge is the maximum response time of its component web services. The QoS distribution of a Parallel Pattern is the distribution of the maximum of independent variables representing component QoS. The distributions be calculated as follows:

$$F(q) = \prod_{i=1}^{n} F_i(q) \quad (4)$$

$$f_(q) = \sum_{i=1}^{n} f_i(q) \prod_{j=1,\ldots,n \& j\neq i} F_j(q) \quad (5)$$

where $f(q)$ and $F(q)$ are the PDF and cumulative distribution function (CDF) of the response time of a Parallel Pattern; $f_i(q)$ and $F_i(q)$ are the PDF and CDF of the response time of component service i; n is the number of component services within this pattern.

3) Conditional Pattern: The response time of a Conditional Pattern is the probability weighted sum of the response time of its component web services. The QoS distribution of a Conditional Pattern can be calculated as follows:

$$f(q) = f_1(q) \odot \ldots \odot f_i(q) \odot \ldots \odot f_n(q) = \sum_{i=1}^{n} p_i f_i(q) \quad (6)$$

where $f(q)$ is the PDF of the response time of a Conditional Pattern; n is the number of component services within this pattern; $f_i(q)$ is the PDF of the QoS of component service i; p_i is the execution probability for component service i.

4) Loop Pattern: In [10], we have given detailed discussion on the structure analysis method for an arbitrary Loop Pattern to compute its QoS. To sum up the method in [10], statistically, a Loop Pattern can be seen as a Conditional Pattern with a Sequential Pattern in each path.

[1] $\delta(x)$ is the Dirac delta function. $\delta(x) = +\infty$ when $x = 0$ and $\delta(x) = 0$ when $x \neq 0$.

With calculation formulae for the execution probability of each path of the Conditional Pattern given in [10] and the formulae of computing the execution time of a Sequential Pattern and Conditional Pattern known (Formulae 3 and 6), the distribution of the response time of a Loop Pattern can be obtained.

5) Computational Complexity of the Proposed QoS Calculation Approach: The computation of convolution takes most time in calculating the QoS of a service composition. Computing convolution directly is normally too slow to be practical. In this paper, with the help of the convolution theorem [13] and the Fast Fourier Transform (FFT) [14], the complexity of the convolution is reduced from $O(n^2)$ to $O(n \log n)$ [13] where n is the number of points in a QoS distribution of a web service in this paper. Therefore, the computational complexity for the proposed QoS calculation approach is $O(mn \log n)$ where m is the number of web services in a service composition and n is the number of discrete points in a QoS distribution.

IV. Evaluation of the Proposed QoS Estimation Approaches

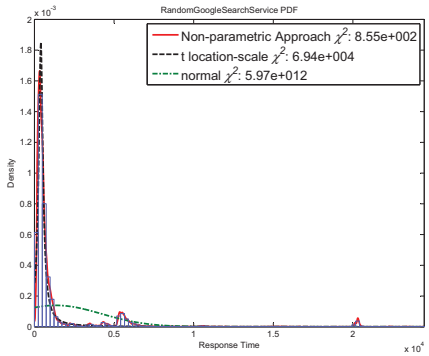
In this section, we will first evaluate the effectiveness of the proposed QoS probability generation method for web services. Then, we will validate the soundness of the proposed QoS estimation method for service compositions.

A. Comparison of QoS Estimation Results for Web Services Based on Different Methods

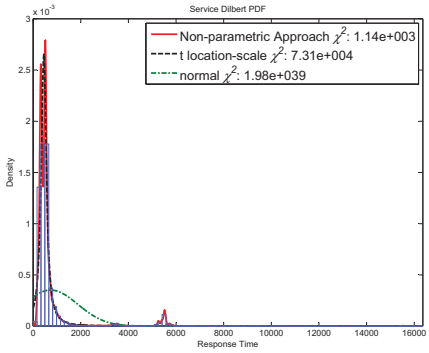
In this subsection,we will compare the two methods of generating QoS distributions for web services. One is the method used in QoSDIST and introduced in section III-A which is referred to as *Non-parametric Approach* in this paper, while the other is fitting QoS sample of a web service into a well known distribution which is commonly used in existing methods and referred to as *Parametric Approach* in this paper. Here in the Parametric Approach, a QoS distribution is assumed to follow a Normal or a T location-scale distribution.

By testing the response time of a web service at a regular time interval, we can get the response time sample (referred to as QoS data $x_1, x_2, \ldots, x_n$ in Section III-A), which is needed in QoS distribution generation. In this paper, the QoS samples of the two web services *Random Image* and *Dilbert* are from WS-DREAM dataset [15].

In Figure 2, histograms represent the QoS samples of web services, solid lines represent QoS probability density distributions generated by Non-parametric Approach, dashed lines represent QoS distributions fitted to T Location-scale distribution, and dash-dotted lines represent QoS distributions fitted to Normal distributions. It can be seen that the solid lines got by Non-parametric Approach fit the QoS sample very well both in the body part and in the tail part. The T Location-scale distributions fit the body part



(a) Random Image PDF



(b) Dilbert PDF

Figure 2. QoS Probability Distributions for Web Services

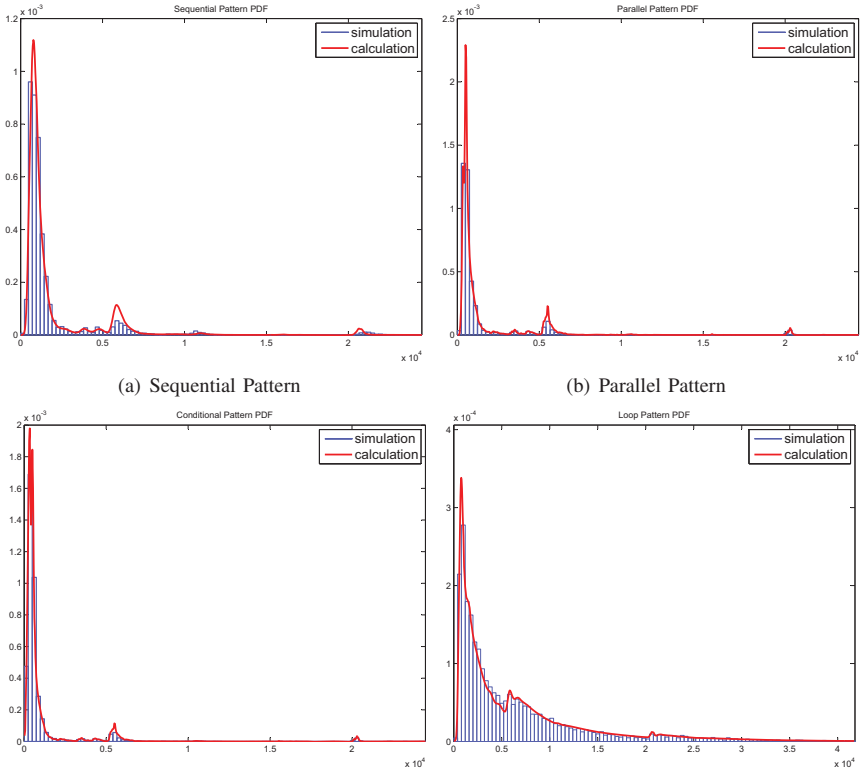
of the QoS sample very well, but not the tail part. As to the Normal distributions, the shape is far from reflecting the real distributions. From the chi-square (χ^2) discrepancy values indicated in Figure 2, it can also be seen that Non-parametric Approach has the smallest discrepancy value to the QoS sample.

To sum up, QoS distributions obtained by Non-parametric Approach are able to represent the real QoS distributions for web services while standard statistical distributions do not have this ability. Therefore, in the next section we will calculate QoS distributions for different composition patterns only based on component QoS distributions generated by Non-parametric approach.

B. Soundness of the Proposed QoS Estimation Approach for Service Compositions

We will run real composite services on web servers and test the QoS probability distributions for these composite services. Then we will use our proposed QoS estimation method to compute the QoS probability distributions for these composite services so that it means that the proposed method is correct if the estimation result by the proposed method comply with the tested QoS probability distributions by experiment.

1) Experiment Setup: In order to consider only the effect of the component QoSs to the composite QoS and ignore



(a) Sequential Pattern (b) Parallel Pattern

(c) Conditional Pattern (d) Loop Pattern

Figure 3. QoS Probability Distributions for Composition Patterns

other factors that may affect the composite QoS, we develop web services on local web servers for testing purpose. In this way, the influence of network performance to a composite service can be ignored so that the tested QoS of a composite service can be used as a standard to validate the correctness of the proposed QoS computation approach for service compositions. Two web services conforming to the non-parametric approach generated QoS probability distributions (see solid lines in Figure 2 (a) and (b)) are developed and deployed on Apache Tomcat 5.5 server. Four BPEL processes, i.e. composite services, are developed and deployed on Active BPEL engine. These composite services are (1) a Sequential Pattern composed of the two developed web services; (2) a Parallel Pattern composed of the two developed web services; (3) a Loop Pattern composed of the two developed web services; and (4) a Conditional Pattern composed of the two developed web services.

Now let us look at how to let a web service execution time follow a specific distribution, i.e. the execution time of a web service per invocation must be a random value conforming to this distribution. A large array of random numbers conforming to the specified distribution is generated and stored in a file. The size of the array is set to 10000. For each execution, the web service will randomly read one value in the file and suspend for the same amount of time as the obtained random value before it sends out a response. By doing so, a web service conforming to certain execution time distribution is developed.

To simulate the transition probabilities in a Conditional Pattern or a Loop Pattern, a random number generator conforming to a uniform distribution is adopted. The transition from one web service to another in a Conditional Pattern or a Loop Pattern is according to the value generated by the random number generator. For example, for a Conditional Pattern with two conditional branches, the execution probability of the first branch is 0.3 and that of the second branch is 0.7. If the value generated by the random number generator for this Conditional Pattern is 0.4 which is larger than 0.3, then this Conditional Pattern will take the second branch at this particular time.

2) Comparison of computation result with experiment result: We invoke each of the four composite services developed in Section IV-B1 for 10000 times and test the response time per invocation. Then we get 10000 response time data for each of the four composite services respectively. The histograms of the testing results are shown in Figure 3 for each of the four composite services.

We use the QoS probability distributions of the web

services developed in Section IV-B1 (see solid lines in Figure 2(a) and (b)) as the component QoSs of the composite services developed in Section IV-B1. The QoS estimation approach proposed in Section III-B is applied to calculate the QoS probability distributions for these composite services. Solid curves in Figure 3 show the calculated QoS probability distributions of the four composite services.

It can be seen from Figure 3 that the calculated QoS distributions based on Non-parametric Approach generated component QoS probability distributions fit the simulation results quite well for all the four composition patterns.

According to the above experimental results and analysis, we can conclude that the proposed QoS estimation approach for service compositions are correct.

V. Implementation of QoSDIST

A. Framework of QoSDIST

Estimating the QoS distributions of a service composition involves large amount of computing. Therefore, the QoS distribution estimation tool QoSDIST adopts the client-server model. The server is in charge of most of the processing and the major workload is allocated on the server end. A Web-based graphical user interface (WebGUI) runs on the client side. Based on this design, a user does not need to install any program to run the GUI so that the system requirement of running this tool is low.

The information exchanged between the clients and the server is based on XML. The input information from WebGUI is encoded into XML by the clients and then sent to the server side. On receiving the XML document, the server side will start processing it and return the processing result as requested in XML.

The framework of QoSDIST is shown in Figure 4 and will be explained next.

`WebGUI`: WebGUI is on the client end of QoSDIST. Through WebGUI, a user can input the process of a service composition and the QoS information for the component services in the service composition (① in Figure 4). The input information will be encoded into XML format and transmitted to the server side of QoSDIST to be processed (⑥ in Figure 4). The output of WebGUI is the QoS distributions for a service composition (⑦ in Figure 4).

The functions of the modules in the QoSDIST server are as follows:

`Translator`: The input of *Translator* is the XML encoded service composition information from client (⑥ in Figure 4). It contains the process information of a service composition and the QoS information for component services. *Translator* decodes the received information and gets a graph structure for the service composition (⑧ in Figure 4).

`Calculator`: The input of *Calculator* are a graph representing the service composition from *Translator* (⑧ in Figure 4) and QoS probability distributions for component services from *QoS Distribution Database*. The QoS probability distributions for a service composition will be calculated here. The calculated QoS probability distributions will be output to *WebGUI* (⑤ in Figure 4).

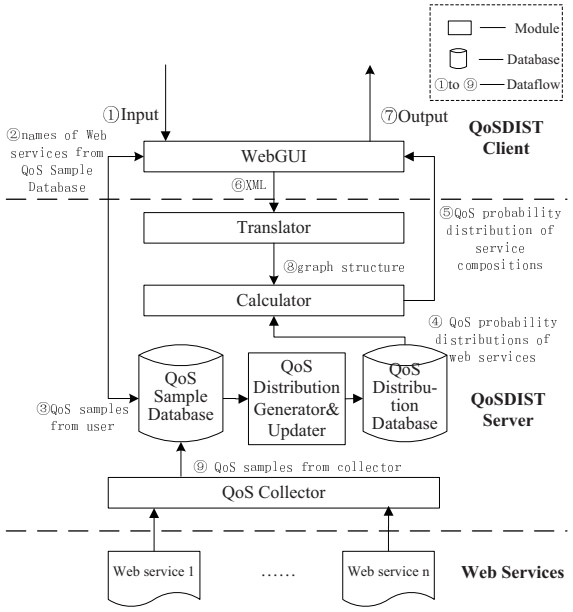


Figure 4. Framework of QoSDIST

`QoS Sample Database`: The information of QoS samples from *QoS Collector* (⑨ in Figure 4) and *WebGUI* (③ in Figure 4) is stored in *QoS Sample Database*. Whenever new samples are coming, these samples will be passed on to *QoS Distribution Generator and Updater*.

`QoS Distribution Generator and Updater`: In *QoS Generator and Updater*, for a web service whose probability distributions can not be found in *QoS Distribution Database*, a QoS probability distribution estimation algorithm will be ran to generate the QoS probability distribution for this web service based on the incoming QoS sample from *QoS Sample Database*. For a web service having its QoS distributions in *QoS Distribution Database*, a QoS probability distribution updating algorithm will be ran to update the QoS probability distribution based on both the original QoS probability distribution in *QoS Distribution Database* and the inputting QoS sample from *QoS Sample Database*.

`QoS Distribution Database`: The information of QoS probability distributions estimated or updated by *QoS Generator and Updater* is stored in *QoS Distribution Database*. The QoS probability distributions are to be used by *Calculator* for composite QoS distribution calculation.

`QoS Collector`: *QoS collector* is in charge of collecting QoS data for web services (⑨ in Figure 4) by testing the QoS of web services at a regular time interval. These

collected QoS data for per QoS metric of a web service is referred to as a QoS sample for that QoS metric of the web service. For example, if $X = 0.024, 0.043, ..., 0.019$ is a set of QoS data collected by `QoS Collector` for the response time of web service *WSExample*, then X is a QoS sample for the response time of *WSExample*. QoS samples collected by *QoS Collector* will be stored at *QoS Sample Database*.

B. An Overview of QoSDIST

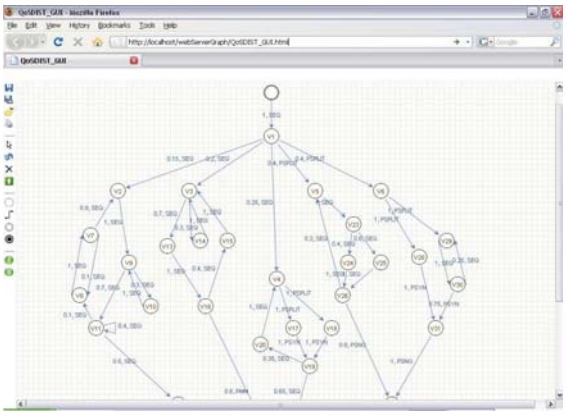
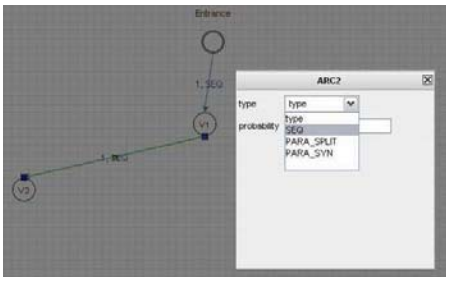


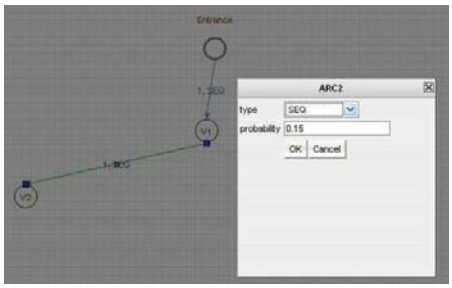
Figure 5. WebGUI of QoSDIST

Figure 5 shows the WebGUI of QoSDIST. A toolbar is on the left side of WebGUI. A user can edit the service graph of a service composition on WebGUI. There are four types of graph elements, i.e. vertex (representing a web service), arc (representing a transition from one web service to another), entrance vertex (representing the entrance web service of a service composition), and exit vertex (representing the exit web service of a service compsoition). For each graph, only one entrance and one exit vertex can be drawn on the WebGUI.

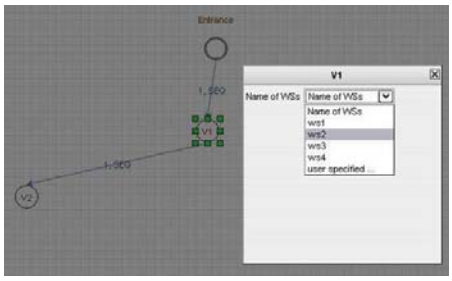
Once an object is drawn on the WebGUI, the property of the object can be edited by double clicking it. Figure 6 shows the property windows of an arc and an vertex. There are three types of arc to select (see Figure 6(a)), which are sequential (SEQ), AND-split (PARA_SPLIT), and AND-join (PARA_SYN). A user can also specify the transition probability of an arc by inputting an value ranging from 0 to 1 in the text area of probability property (see Figure 6(b)). A user can select per vertex a web service whose QoS is either available in QoSDIST or can be inputted by the user by selecting the *user specified* option in the vertex property window (see Figure 6(c)). The names of the web services in Figure 6(c) are from the QoSDIST server side. When the WebGUI is initialized, the client side has a communication with the server side to obtain the names of web services. These names are to be selected for each component service of a service composition by the user (see Figure 6(c)). For those component services



(a) Editing type property of transition ARC2



(b) Editing probability property of transition ARC2



(c) Editing web service property of service V1

Figure 6. Editing properties of a service composition

that can not find a matching web service, a user can specify the QoS for that component service by selecting the *user specified* option in the vertex property window and inputting QoS samples (see Figure 6(c)). These samples (③ in Figure 4) will be transmitted to and stored on the server.

After drawing a graph on WebGUI (see the graph on WebGUI in Figure 5), a user can send this graph to the server side and get the calculation result. The calculation result will be shown through a plot on a pop up window of WebGUI (see Figure 7). This plot is a probability density function distribution. X-axis represents the specific QoS metric. Y-axis represents probability density. A user can get the exact probability density value of a particular point by putting the mouse over that point.

VI. Conclusion

In this paper, we introduced the design and realization of QoSDIST: a QoS distribution estimation tool. QoSDIST can estimate QoS probability density distributions for service compositions. Two ideas in QoSDIST are completely new:

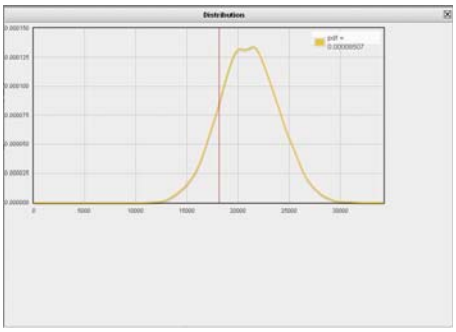


Figure 7. Result Display

(1) Non-parametric QoS estimation approach is adopted to estimate QoS distributions for web services based on tested QoS samples. The virtue of Non-parametric QoS estimation approach is that it can estimate QoS distributions much more accurately than Parametric Approach which is used in estimating QoS distributions for web services by esiting methods. (2) Formulae are designed to calculate QoS distributions for service compositions. The advantage of using formulae in calculating QoS is the flexibility in QoS representation, which means the proposed method can do the estimation with the QoS being represented in any forms, i.e. single values, probability mass functions, or probability distributions .

The current version of QoSDIST is a prototype system. The following work needs to be done in the future:

- Fault tolerance ability will be enhanced. For example, an concurrent split arc must finally be followed by a synchronized merge or a single merge arc.
- QoSDIST will finally be developed into a system which is able to estimate QoS distributions in real time. Some applications may involve a large number of web services and need quick response. Therefore, having the capability of real time QoS estimation is meaningful.
- Multi-user support and multi-browser support will be added.

REFERENCES

[1] K. Lee, J. Jeon, W. Lee, S.-H. Jeong, and S.-W. Park, "Qos for web services: Requirements and possible approaches," W3C Working Group, Tech. Rep. 25, November 2003.

[2] L. Zeng, B. Benatallah, A. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "Qos-aware middleware for web services composition," *Software Engineering, IEEE Transactions on*, vol. 30, no. 5, pp. 311–327, May 2004.

[3] D. Ardagna and B. Pernici, "Adaptive service composition in flexible processes," *IEEE Transactions on Software Engineering*, vol. 33, no. 6, pp. 369–384, June 2007.

[4] G. Canfora, M. Di Penta, R. Esposito, and M. L. Villani, "Qos-aware replanning of composite web services," in *Proceedings of the IEEE International Conference on Web Services (ICWS)*, USA, 2005.

[5] A. Klein, F. Ishikawa, and B. Bauer, "A probabilistic approach to service selection with conditional contracts and usage patterns," in *ICSOC-ServiceWave '09: Proceedings of the 7th International Joint Conference on Service-Oriented Computing*. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 253–268.

[6] S. Rosario, A. Benveniste, S. Haar, and C. Jard, "Probabilistic qos and soft contracts for transaction-based web services orchestrations," *IEEE Transactions on Services Computing*, vol. 1, no. 4, pp. 187–200, 2008.

[7] C. Hughes and J. Hillman, "Qos explorer: A tool for exploring qos in composed services," in *IEEE International Conference on Web Services (ICWS)*, USA, 2006.

[8] J. Cardoso, J. Miller, A. Sheth, and J. Arnold, "Quality of service for workflows and web service processes," *Journal of Web Semantics*, vol. 1, pp. 281–308, 2004.

[9] S.-Y. Hwang, H. Wang, J. Tang, and J. Srivastava, "A probabilistic approach to modeling and estimating the qos of web-services-based workflows," *Information Sciences*, vol. 177, no. 23, pp. 5484–5503, 2007.

[10] H. Zheng, W. Zhao, J. Yang, and A. Bouguettaya, "Qos analysis for web service composition," in *IEEE International Conference on Services Computing (SCC)*, India, 2009.

[11] B. Silverman, *Density Estimation for Statistics and Data Analysis*, 1st ed. Chapman and Hall, 1986.

[12] H. Zheng, J. Yang, and W. Zhao, "Qos analysis and service selection for composite services," in *in Proceedings of the 7th IEEE International Conference on Services Computing (SCC2010)*, Miami, Florida, USA, July 2010.

[13] http://en.wikipedia.org/wiki/Convolution_theorem.

[14] http://en.wikipedia.org/wiki/Fast_Fourier_transform.

[15] Z. Zheng, H. Ma, M. R. Lyu, and I. King, "Wsrec: A collaborative filtering based web service recommender system," in *in Proceedings of the 7th IEEE International Conference on Web Services (ICWS2009)*, Los Angeles, CA, USA, July 2009.